

1. Introduction
2. Basic tag
3. Meta Tag
4. Comments
5. Image Tag
6. Tables
7. Lists
8. HTML Frames
9. HTML Marquee
10. **Introduction**

HTML stands for **H**yper**t**ext **M**arkup **L**anguage, and it is the most widely used language to write Web Pages.

Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus the link available on a webpage is called Hypertext.

As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

**Basic HTML Document**

In its simplest form, following is an example of an HTML document:

<!DOCTYPE html>

<html>

<head>

<title>This is document title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>Document content goes here.....</p>

</body>

</html>

1. **Basic Tag**

**Heading Tags**

Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements <h1>, <h2>, <h3>, <h4>, <h5>, and <h6**>**. While displaying any heading, browser adds one line before and one line after that heading.

**Example**

<!DOCTYPE html>

<html>

<head>

<title>Heading Example</title>

</head>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

<h4>This is heading 4</h4>

<h5>This is heading 5</h5>

<h6>This is heading 6</h6>

</body>

</html>

**Paragraph Tag**

The **<p>** tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening <p> and a closing </p> tag as shown below in the example:

**Example**

<!DOCTYPE html>

<html>

<head>

<title>Paragraph Example</title>

</head>

<body>

<p>Here is a first paragraph of text. </p>

<p>Here is a second paragraph of text. </p>

<p>Here is a third paragraph of text. </p>

</body>

</html>

**Core**[**Attributes**](http://www.tutorialspoint.com/)

The four core [attributes](http://www.tutorialspoint.com/) that can be used on the majority of [HTML](http://www.tutorialspoint.com/)elements (although not all) are:

1. id
2. title
3. class
4. style
5. **The id Attribute**

The **id** attribute of an [HTML](http://www.tutorialspoint.com/) tag can be used to uniquely identify any element within an [HTML](http://www.tutorialspoint.com/) page. There are two primary reasons that you might want to use an id attribute on an element:

If an element carries an id attribute as a unique identifier it is possible to identify just that element and its content.

If you have two elements of the same name within a Web page (or [style sheet](http://www.tutorialspoint.com/)), you can use the id attribute to distinguish between elements that have the same name.

We will discuss [style sheet](http://www.tutorialspoint.com/) in separate [tutorial](http://www.tutorialspoint.com/). For now, let's use the id attribute to distinguish between two paragraph elements as shown below.

**Example**

<p id="[html](http://www.tutorialspoint.com/)">This Para explains what is [HTML](http://www.tutorialspoint.com/)</p>

<p id="css">This Para explains what is Cascading [Style Sheet](http://www.tutorialspoint.com/)</p>

1. **The title Attribute**

The **title** attribute gives a suggested title for the element. They syntax for the **title** attribute is similar as explained for **id** attribute:

The behaviour of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.

**Example:**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>The title Attribute Example</title>

</head>

<body>

<h3 title="Hello [HTML](http://www.tutorialspoint.com/)!">Titled Heading Tag Example</h3>

</body>

</[html](http://www.tutorialspoint.com/)>

Titled Heading Tag Example

Now try to bring your cursor over "Titled Heading Tag Example" and you will see that whatever title you used in your code is coming out as a tooltip of the cursor.

The class Attribute

The **class** attribute is used to associate an element with a [style sheet](http://www.tutorialspoint.com/), and specifies the class of element. You will learn more about the use of the class attribute when you will learn Cascading [Style Sheet](http://www.tutorialspoint.com/) (CSS). So for now you can avoid it.

The value of the attribute may also be a space-separated list of class names. For example:

Class="className1 className2 className3"

The style Attribute

The style attribute allows you to specify Cascading [Style Sheet](http://www.tutorialspoint.com/) (CSS) rules within the element.

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>The style Attribute</title>

</head>

<body>

<p style="font-family:arial; color:#FF0000;">Some text...</p>

</body>

</[html](http://www.tutorialspoint.com/)>

This will produce following result:

Some text...

1. **Meta Tag**

HTML lets you specify metadata - additional important information about a document in a variety of ways. The META elements can be used to include name/value pairs describing properties of the HTML document, such as author, expiry date, a list of keywords, document author etc.

The <Meta> tag is used to provide such additional information. This tag is an empty element and so does not have a closing tag but it carries information within its attributes.

You can include one or more meta tags in your document based on what information you want to keep in your document but in general, meta tags do not impact physical appearance of the document so from appearance point of view, it does not matter if you include them or not.

Adding Meta Tags to Your Documents

You can add metadata to your web pages by placing <meta> tags inside the header of the document which is represented by **<head>** and **</head>** tags. A meta tag can have following attributes in addition to core attributes:

1. Specifying Keywords

<!DOCTYPE html>

<html>

<head>

<title>Meta Tags Example</title>

<meta name="keywords" content="HTML, Meta Tags, Metadata" />

</head>

<body>

<p>Hello HTML5!</p>

</body>

</html>

1. Document Description

<!DOCTYPE html>

<html>

<head>

<title>Meta Tags Example</title>

<meta name="keywords" content="HTML, Meta Tags, Metadata" />

<meta name="description" content="Learning about Meta Tags." />

</head>

<body>

<p>Hello HTML5!</p>

</body>

</html>

1. Document Refreshing

<!DOCTYPE html>

<html>

<head>

<title>Meta Tags Example</title>

<meta name="keywords" content="HTML, Meta Tags, Metadata" />

<meta name="description" content="Learning about Meta Tags." />

<meta name="revised" content="Enosislearning, 3/7/2014" />

<meta http-equiv="refresh" content="5" />

</head>

<body>

<p>Hello HTML5!</p>

</body>

</html>

1. Page Redirection

<!DOCTYPE html>

<html>

<head>

<title>Meta Tags Example</title>

<meta name="keywords" content="HTML, Meta Tags, Metadata" />

<meta name="description" content="Learning about Meta Tags." />

<meta name="revised" content="Enosislearning, 3/7/2014" />

<meta http-equiv="refresh" content="5; url=http://www.enosislearning.com" />

</head>

<body>

<p>Hello HTML5!</p>

</body>

</html>

1. Setting Cookies

<!DOCTYPE html>

<html>

<head>

<title>Meta Tags Example</title>

<meta name="keywords" content="HTML, Meta Tags, Metadata" />

<meta name="description" content="Learning about Meta Tags." />

<meta name="revised" content="Enosislearning, 3/7/2014" />

<meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15 23:59:59 GMT;" />

</head>

<body>

<p>Hello HTML5!</p>

</body>

</html>

**4) Comment**

HTML comments are placed in between **<!-- ... -->** tags. So any content placed with-in <!-- ... --> tags will be treated as comment and will be completely ignored by the browser.

Example

<!DOCTYPE html>

<html>

<head> <!-- Document Header Starts -->

<title>This is document title</title>

</head> <!-- Document Header Ends -->

<body>

<p>Document content goes here.....</p>

</body>

</html>

**5) Image Tag**

You can insert any image in your web page by using <img> tag. Following is the simple syntax to use this tag.

<img src="Image URL" ... [attributes](http://www.tutorialspoint.com/)-list/>

The <img> tag is an empty tag, which means that it can contain only list of attributes and it has no closing tag.

Example

To try following example, let's keep our [HTML](http://www.tutorialspoint.com/) file test.htm and image file test.png in the same directory:

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>Using Image in Webpage</title>

</head>

<body>

<p>[Simple](http://www.tutorialspoint.com/) Image Insert</p>

<img src="/[html](http://www.tutorialspoint.com/)/[images](http://www.tutorialspoint.com/)/test.png" alt="Test Image" title=””/>

</body>

</[html](http://www.tutorialspoint.com/)>

**6) Tables**

The [HTML tables](http://www.tutorialspoint.com/) allow web authors to arrange data like text, [images](http://www.tutorialspoint.com/), [links](http://www.tutorialspoint.com/), other [tables](http://www.tutorialspoint.com/), etc. into rows and columns of cells.

The [HTML tables](http://www.tutorialspoint.com/) are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells.

Example

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML Tables](http://www.tutorialspoint.com/)</title>

</head>

<body>

<table border="1">

<tr>

<td>Row 1, Column 1</td>

<td>Row 1, Column 2</td>

</tr>

<tr>

<td>Row 2, Column 1</td>

<td>Row 2, Column 2</td>

</tr>

</table>

</body>

</[html](http://www.tutorialspoint.com/)>

This will produce following result:

|  |  |
| --- | --- |
| Row 1, Column 1 | Row 1, Column 2 |
| Row 2, Column 1 | Row 2, Column 2 |

Here border is an attribute of <table> tag and it is used to put a border across all the cells. If you do not need a border then you can use border="0".

Table Heading

Table heading can be defined using **<th>** tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Normally you will put your top row as table heading as shown below, otherwise you can use <th> element in any row.

Example

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) Table Header</title>

</head>

<body>

<table border="1">

<tr>

<th>Name</th>

<th>Salary</th>

</tr>

<tr>

<td>Ramesh Raman</td>

<td>5000</td>

</tr>

<tr>

<td>Shabbir Hussein</td>

<td>7000</td>

</tr>

</table>

</body>

</[html](http://www.tutorialspoint.com/)>

This will produce following result:

|  |  |
| --- | --- |
| **Name** | **Salary** |
| Ramesh Raman | 5000 |
| Shabbir Hussein | 7000 |

**Cellpadding and Cellspacing**[**Attributes**](http://www.tutorialspoint.com/)

There are two attribiutes called *cellpadding* and *cellspacing* which you will use to adjust the white space in your table cells. The cellspacing attribute defines the width of the border, while cellpadding represents the distance between cell borders and the content within a cell.

**Example**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) Table Cellpadding</title>

</head>

<body>

<table border="1" cellpadding="5" cellspacing="5">

<tr>

<th>Name</th>

<th>Salary</th>

</tr>

<tr>

<td>Ramesh Raman</td>

<td>5000</td>

</tr>

<tr>

<td>Shabbir Hussein</td>

<td>7000</td>

</tr>

</table>

</body>

</[html](http://www.tutorialspoint.com/)>

This will produce following result:

|  |  |
| --- | --- |
| **Name** | **Salary** |
| Ramesh Raman | 5000 |
| Shabbir Hussein | 7000 |

**Colspan and Rowspan**[**Attributes**](http://www.tutorialspoint.com/)

You will use **colspan** attribute if you want to merge two or more columns into a single column. Similar way you will use **rowspan** if you want to merge two or more rows.

**Example**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) Table Colspan/Rowspan</title>

</head>

<body>

<table border="1">

<tr>

<th>Column 1</th>

<th>Column 2</th>

<th>Column 3</th>

</tr>

<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>

<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>

<tr><td colspan="3">Row 3 Cell 1</td></tr>

</table>

</body>

</[html](http://www.tutorialspoint.com/)>

|  |  |  |
| --- | --- | --- |
| **Column 1** | **Column 2** | **Column 3** |
| Row 1 Cell 1 | Row 1 Cell 2 | Row 1 Cell 3 |
| Row 2 Cell 2 | Row 2 Cell 3 |
| Row 3 Cell 1 | | |

**7) Lists**

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

* **<ul>** - An unordered list. This will list items using plain bullets.
* **<ol>** - An ordered list. This will use different schemes of numbers to list your items.

## HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML **<ul>** tag. Each item in the list is marked with a bullet.

Example

<!DOCTYPE html>

<html>

<head>

<title>HTML Unordered List</title>

</head>

<body>

<ul>

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ul>

</body>

</html>

This will produce following result:

* Beetroot
* Ginger
* Potato
* Radish
* The type Attribute

You can use type attribute for <ul> tag to specify the type of bullet you like. By default it is a disc. Following are the possible options:

1. <ul type="square">
2. <ul type="disc">
3. <ul type="circle">

HTML Ordered Lists

If you are required to put your items in a numbered list instead of bulleted then HTML ordered list will be used. This list is created by using <ol> tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>.

Example

<!DOCTYPE html>

<html>

<head>

<title>HTML Ordered List</title>

</head>

<body>

<ol>

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ol>

</body>

</html>

This will produce following result:

1. Beetroot
2. Ginger
3. Potato
4. Radish
5. The type Attribute

You can use type attribute for <ol> tag to specify the type of numbering you like. By default it is a number. Following are the possible options:

1. <ol type="1"> - Default-Case Numerals.
2. <ol type="I"> - Upper-Case Numerals.
3. <ol type="i"> - Lower-Case Numerals.
4. <ol type="a"> - Lower-Case Letters.
5. <ol type="A"> - Upper-Case Letters.

**8) Frames**

[HTML frames](http://www.tutorialspoint.com/) are used to divide your browser window into multiple sections where each section can load a separate [HTML](http://www.tutorialspoint.com/) document. A collection of [frames](http://www.tutorialspoint.com/) in the browser window is known as a frameset. The window is divided into [frames](http://www.tutorialspoint.com/) in a similar way the [tables](http://www.tutorialspoint.com/) are organized: into rows and columns.

**Disadvantages of**[**Frames**](http://www.tutorialspoint.com/)

There are few drawbacks with using [frames](http://www.tutorialspoint.com/), so it's never recommended to use[frames](http://www.tutorialspoint.com/) in your webpages:

Some smaller devices cannot cope with [frames](http://www.tutorialspoint.com/) often because their screen is not big enough to be divided up.

Sometimes your page will be displayed differently on different computers due to different screen resolution.

The browser's *back button* might not work as the user hopes.

There are still few browsers that do not support frame technology.

**Browser Support for**[**Frames**](http://www.tutorialspoint.com/)

If a user is using any old browser or any browser which does not support [frames](http://www.tutorialspoint.com/) then <noframes> element should be displayed to the user.

So you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.

You can put some nice message for your user having old browsers. For example *Sorry!! your browser does not support*[*frames*](http://www.tutorialspoint.com/)*.* as shown in the above example.

**Frame's name and target**[**attributes**](http://www.tutorialspoint.com/)

One of the most popular uses of [frames](http://www.tutorialspoint.com/) is to place navigation bars in one frame and then load main pages into a separate frame.

Let's see following example where a test.htm file has following code:

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) Target [Frames](http://www.tutorialspoint.com/)</title>

</head>

<frameset cols="200, \*">

<frame src="/[html](http://www.tutorialspoint.com/)/menu.htm" name="menu\_page" />

<frame src="/[html](http://www.tutorialspoint.com/)/main.htm" name="main\_page" />

<noframes>

<body>

Your browser does not support [frames](http://www.tutorialspoint.com/).

</body>

</noframes>

</frameset>

</[html](http://www.tutorialspoint.com/)>

Here we have created two columns to fill with two [frames](http://www.tutorialspoint.com/). The first frame is 200 pixels wide and will contain the navigation menubar implemented by **menu.htm** file. The second column fills in remaining space and will contain the main part of the page and it is implemented by **main.htm** file. For all the three [links](http://www.tutorialspoint.com/) available in menubar, we have mentioned target frame as **main\_page**, so whenever you click any of the[links](http://www.tutorialspoint.com/) in menubar, available link will open in main\_page.

Following is the content of menu.htm file

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<body bgcolor="#4a7d49">

<a href="http://www.google.com" target="main\_page">Google</a>

<br /><br />

<a href="http://www.microsoft.com" target="main\_page">Microsoft</a>

<br /><br />

<a href="http://news.bbc.co.uk" target="main\_page">BBC News</a>

</body>

</[html](http://www.tutorialspoint.com/)>

Following is the content of main.htm file:

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<body bgcolor="#b5dcb3">

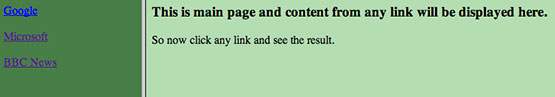
<h3>This is main page and content from any link will be displayed here.</h3>

<p>So now click any link and see the result.</p>

</body>

</[html](http://www.tutorialspoint.com/)>

When we load **test.htm** file, it produces following result:



Now you can try to click [links](http://www.tutorialspoint.com/) available in the left panel and see the result. The *target*attribute can also take one of the following values:

|  |  |
| --- | --- |
| Option | Description |
| \_self | Loads the page into the current frame. |
| \_blank | Loads a page into a new browser window.opening a new window. |
| \_parent | Loads the page into the parent window, which in the case of a single frameset is the main browser window. |
| \_top | Loads the page into the browser window, replacing any current [**frames**](http://www.tutorialspoint.com/). |
| targetframe | Loads the page into a named targetframe. |

**9) Marquee**

An [HTML](http://www.tutorialspoint.com/) marquee is a scrolling piece of text displayed either horizontally across or vertically down your webpage depending on the settings. This is created by using[HTML](http://www.tutorialspoint.com/) <[marquees](http://www.tutorialspoint.com/)> tag.

**Note:**The [HTML](http://www.tutorialspoint.com/) <marquee> tag may not be supported by various browsers so its not recommended to rely on this tag, instead you can use Javascript and CSS to create such effects.

**Syntax**

A [simple](http://www.tutorialspoint.com/) syntax to use [HTML](http://www.tutorialspoint.com/) <marquee> tag is as follows:

<marquee attribute\_name="attribute\_value"....more [attributes](http://www.tutorialspoint.com/)>

One or more lines or text message or image

</marquee>

**The <marquee> Tag**[**Attributes**](http://www.tutorialspoint.com/)

Following is the list of important [attributes](http://www.tutorialspoint.com/) which can be used with <marquee> tag.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| width | This specifies the width of the marquee. This can be a value like 10 or 20% etc. |
| height | This specifies the height of the marquee. This can be a value like 10 or 20% etc. |
| direction | This specifies the direction in which marquee should scroll. This can be a value like *up*, *down*, *left* or *right*. |
| behavior | This specifies the type of scrolling of the marquee. This can have a value like*scroll*, *slide* and *alternate*. |
| scrolldelay | This specifies how long to delay between each jump. This will have a value like 10 etc. |
| scrollamount | This specifies the speed of marquee text. This can have a value like 10 etc. |
| loop | This specifies how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly. |
| bgcolor | This specifies background color in terms of color name or color hex value. |
| hspace | This specifies horizontal space around the marquee. This can be a value like 10 or 20% etc. |
| vspace | This specifies vertical space around the marquee. This can be a value like 10 or 20% etc. |

Below are few examples to demonstrate the usage of marquee tag.

**Examples - 1**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) marquee Tag</title>

</head>

<body>

<marquee>This is basic example of marquee</marquee>

</body>

</[html](http://www.tutorialspoint.com/)>

**Examples - 2**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) marquee Tag</title>

</head>

<body>

<marquee width="50%">This example will take only 50% width</marquee>

</body>

</[html](http://www.tutorialspoint.com/)>

**Examples - 3**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) marquee Tag</title>

</head>

<body>

<marquee direction="right">This text will scroll from left to right</marquee>

</body>

</[html](http://www.tutorialspoint.com/)>

**Examples - 4**

<!DOCTYPE [html](http://www.tutorialspoint.com/)>

<[html](http://www.tutorialspoint.com/)>

<head>

<title>[HTML](http://www.tutorialspoint.com/) marquee Tag</title>

</head>

<body>

<marquee direction="up">This text will scroll from bottom to up</marquee>

</body>

</[html](http://www.tutorialspoint.com/)>



1. Introduction
2. Selectors
3. Box Model
4. Backgrounds
5. Border
6. Gradient
7. Drop Shadows
8. 2D Transformations
9. 3D Transformations
10. Transition
11. Animations
12. **Introduction**

There's no difference. Just like HTML and HTML 5. CSS 3 is simply the latest 'version' of CSS which has more advanced features than earlier 'releases'.

1. **Selectors**

**What is Selector?**

Selectors are one of the most important aspects of CSS as they are used to select elements on a web page so that they can be styled. You can define selectors in various ways.

* Universal Selector
* Element Type Selector
* Id Selectors
* Class Selectors
* Descendant Selectors
* Child Selectors
* Grouping Selectors

1. **Universal Selector**

\*{

Margin:0;

Padding:0

}

1. **Element Type Selector**

p{color: blue;}

1. **Id Selectors**

<style type="text/css">

#error {

color: #ff0000;

}

</style>

<p id="error"></p>

1. **Class Selectors**

<style type="text/css">

.error {

color: #ff0000;

}

</style>

<p class="error">This is warning</p>

1. **Descendant Selector**

<style type="text/css">

h1 em {color: green;}

ul.menu {padding: 0; list-style: none; }

ul.menu li{ display: inline; }

ul.menu li a {margin: 10px; text-decoration: none; }

</style>

<h1>This is <em>Heading </em></h1>

<ul class="menu">

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Services </a></li>

<li><a href="#">Contact </a></li>

</ul>

1. Child Selectors

<style type="text/css">

ul > li {

list-style: square;

}

ul > li ol {

list-style: none;

}

</style>

<h1>This is<em>Heading </em></h1>

<ul>

<li><a href="#">Home </a></li>

<li><a href="#">About </a></li>

<li>

<a href="#">Services </a>

<ol>

<li><a href="#">Design </a></li>

<li><a href="#">Development </a></li>

</ol>

</li>

<li><a href="#">Contact </a></li></ul>

1. **Grouping Selectors**

<style type="text/css">

h1, h2, h3 {font-weight: normal; }

h1 {font-size: 36px; }

h2 {font-size: 28px; }

h3 {font-size: 22px; }

</style>

<h1>This heading h1</h1>

<h2>This heading h2</h2>

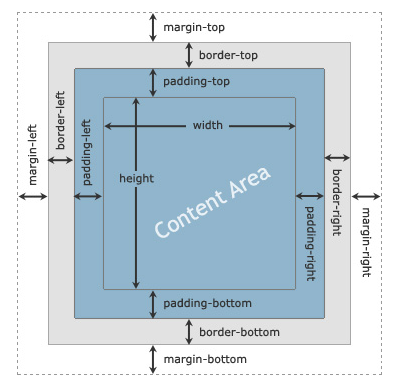
<h3>This heading h3</h3>

1. **Box Model**

The CSS box model is essentially a box that wraps around HTML elements, and it consists of

* margins
* borders
* padding
* the actual content

## WHAT IS THE BOX MODEL?



1. **Background**

The CSS3 provides several new properties to manipulate the background of an element like background clipping, multiple backgrounds, and the option to adjust the background size.

CSS3 gives you ability to add multiple backgrounds to a single element. The backgrounds are layered on the top of one another. The number of layers is determined by the number of comma-separated values in the background-image or background shorthand property.

**Example:**

<html lang="en"><head>

<title>Example of CSS3 Multiple Backgrounds</title>

<style type="text/css">

.box {

width: 100%;

height: 500px;

background:url("images/birds.png") no-repeat center, url("images/clouds.png") repeat-x center left 109px, url("images/sun.png") no-repeat 10% 30%, #BDDDF9;

}

</style>

</head>

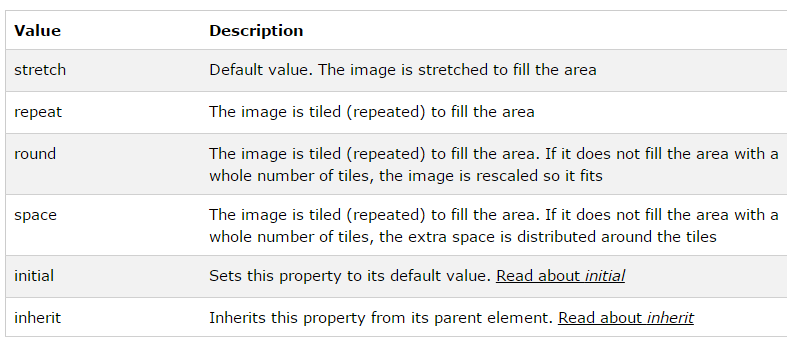
<body>

<div class="box"></div>

</body></html>

1. **Border**

The CSS3 provides two new properties for styling the borders of an element in a more elegant way — the border-image property for adding the images to borders, and the border-radius property for making the rounded corners without using any images.



**Example:**

<html lang="en"><head>

<title>Example of CSS3 Border Images</title>

<style type="text/css">

.box {

width: 300px;

height: 150px;

border: 15px solid transparent;

-webkit-border-image: url("images/border.png") 30 30 round; /\* Safari 3.1-5 \*/

-o-border-image: url("images/border.png") 30 30 round; /\* Opera 11-12.1 \*/

border-image: url("images/border.png") 30 30 round;

}

</style>

</head>

<body>

<div class="box"></div>

</body></html>

**Example:**

<html lang="en"><head>

<title>Example of CSS3 Border Images</title>

<style type="text/css">

.box {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

-webkit-border-image: url("images/border.png") 30 30 round; /\* Safari 3.1-5 \*/

-o-border-image: url("images/border.png") 30 30 round; /\* Opera 11-12.1 \*/

border-image: url("images/border.png") 30 30 round;

}

.box1 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 50 repeat;

}

.box2 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 20% round;

}

.box3 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 30% round;

}

.box4 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 40% round;

}

.box5 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 20% stretch;

}

.box6 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 30% stretch;

}

.box7 {

width: 300px;

height: 150px;

float:left;

margin-left:50px;

border: 15px solid transparent;

border-image: url("images/border.png") 40% stretch;

}

</style>

</head>

<body>

<div class="box"></div>

<div class="box1"></div>

<div class="box2"></div>

<div class="box3"></div>

<div class="box4"></div>

<div class="box5"></div>

<div class="box6"></div>

<div class="box7"></div>

</body></html>

1. **Gradient**

**Types of Gradient are –**

* Linear Gradient
* Radial Gradient

1. **Linear Gradient**
   1. **Linear Gradient - Top to Bottom**

<html lang="en"><head>

<title>Example of Linear Gradients from Left to Right</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-linear-gradient( red, yellow);

/\* For Internet Explorer 10 \*/

background: -ms-linear-gradient( red, yellow);

/\* Standard syntax \*/

background: linear-gradient(, red, yellow);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

* 1. **Linear Gradient - Left to Right**

<html lang="en"><head>

<title>Example of Linear Gradients from Left to Right</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-linear-gradient(left, red, yellow);

/\* For Internet Explorer 10 \*/

background: -ms-linear-gradient(left, red, yellow);

/\* Standard syntax \*/

background: linear-gradient(to right, red, yellow);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

* 1. **Linear Gradient – Diagonal**

<html lang="en"><head>

<title>Example of Linear Gradients along Diagonal</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-linear-gradient(bottom left, red, yellow);

/\* For Internet Explorer 10 \*/

background: -ms-linear-gradient(bottom left, red, yellow);

/\* Standard syntax \*/

background: linear-gradient(to top right, red, yellow);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

* 1. **Setting Direction of Linear Gradients Using Angles**

<html lang="en"><head>

<title>Example of Setting Linear Gradients Direction Using Angles</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Permalink - use to edit and share this gradient: http://colorzilla.com/gradient-editor/#184663+0,c97c38+100 \*/

background: #184663; /\* Old browsers \*/

background: -moz-linear-gradient(-45deg, #184663 0%, #c97c38 100%); /\* FF3.6+ \*/

background: -webkit-gradient(linear, left top, right bottom, color-stop(0%,#184663), color-stop(100%,#c97c38)); /\* Chrome,Safari4+ \*/

background: -webkit-linear-gradient(-45deg, #184663 0%,#c97c38 100%); /\* Chrome10+,Safari5.1+ \*/

background: -o-linear-gradient(-45deg, #184663 0%,#c97c38 100%); /\* Opera 11.10+ \*/

background: -ms-linear-gradient(-45deg, #184663 0%,#c97c38 100%); /\* IE10+ \*/

background: linear-gradient(135deg, #184663 0%,#c97c38 100%); /\* W3C \*/

filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#184663', endColorstr='#c97c38',GradientType=1 ); /\* IE6-9 fallback on horizontal gradient \*/

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

* 1. **Creating Linear Gradients Using Multiple Color Stops**

<html lang="en"><head>

<title>Example of Linear Gradients with Multiple Color Stops</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-linear-gradient(red, yellow, lime);

/\* For Internet Explorer 10 \*/

background: -ms-linear-gradient(red, yellow, lime);

/\* Standard syntax \*/

background: linear-gradient(red, yellow, lime);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

* 1. **Setting the Location Color Stops**

<html lang="en"><head>

<title>Example of Setting the Color Stops Location</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-linear-gradient(red, yellow 30%, lime 60%);

/\* For Internet Explorer 10 \*/

background: -ms-linear-gradient(red, yellow 30%, lime 60%);

/\* Standard syntax \*/

background: linear-gradient(red, yellow 30%, lime 60%);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

* 1. **Repeating the Linear Gradients**

<html lang="en"><head>

<title>Example of Setting the Color Stops Location</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-linear-gradient(red, yellow 30%, lime 60%);

/\* For Internet Explorer 10 \*/

background: -ms-linear-gradient(red, yellow 30%, lime 60%);

/\* Standard syntax \*/

background: linear-gradient(red, yellow 30%, lime 60%);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

1. **Radial Gradient**
2. **Creating CSS3 Radial Gradients**

<html lang="en"><head>

<title>Example of Radial Gradients</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-radial-gradient(red, yellow, lime);

/\* For Internet Explorer 10 \*/

background: -ms-radial-gradient(red, yellow, lime);

/\* Standard syntax \*/

background: radial-gradient(red, yellow, lime);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

1. **Setting the Shape of Radial Gradients**

<html lang="en"><head>

<title>Example of Setting the Shape Radial Gradients</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-radial-gradient(circle, red, yellow, lime);

/\* For Internet Explorer 10 \*/

background: -ms-radial-gradient(circle, red, yellow, lime);

/\* Standard syntax \*/

background: radial-gradient(circle, red, yellow, lime);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

1. **Setting the Size of Radial Gradients**

<html lang="en"><head>

<title>Example of Setting the Size Radial Gradients</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: red;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-radial-gradient(left bottom, circle farthest-side, red, yellow, lime);

/\* For Internet Explorer 10 \*/

background: -ms-radial-gradient(left bottom, circle farthest-side, red, yellow, lime);

/\* Standard syntax \*/

background: radial-gradient(circle farthest-side at left bottom, red, yellow, lime);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

1. **Repeating the Radial Gradients**

<html lang="en"><head>

<title>Example of Repeating the Size Radial Gradients</title>

<style type="text/css">

.gradient {

width: 400px;

height: 300px;

/\* Fallback for browsers that don't support gradients \*/

background: white;

/\* For Safari 5.1 to 6.0 \*/

background: -webkit-repeating-radial-gradient(black, white 10%, lime 20%);

/\* For Internet Explorer 10 \*/

background: -ms-repeating-radial-gradient(black, white 10%, lime 20%);

/\* Standard syntax \*/

background: repeating-radial-gradient(black, white 10%, lime 20%);

}

</style>

</head>

<body>

<div class="gradient"></div>

</body></html>

1. **Box Shadow**

The box-shadow property can be used to add shadow to the element's boxes. You can even apply more than one shadow effects using a comma-separated list of shadows. The basic syntax of creating a box shadow can be given with:

box-shadow: offset-x offset-y blur-radius color;

Example:

1. **Single Color Shadow**

<html lang="en"><head>

<title>Example of CSS3 Box Shadow Effect</title>

<style type="text/css">

.box{

width: 200px;

height: 150px;

background: #ccc;

box-shadow: 50px 50px 50px red;

}

</style>

</head>

<body>

<div class="box"></div>

</body></html>

1. **Multiple Color Shadow**

<html lang="en"><head>

<title>Example of CSS3 Multiple Box Shadow Effects</title>

<style type="text/css">

.box{

width: 200px;

height: 150px;

background: #000;

box-shadow: 5px 5px 10px red, 10px 10px 20px yellow;

}

</style>

</head>

<body>

<div class="box"></div>

</body></html>

1. **Text Shadow**

<html lang="en"><head>

<title>Example of CSS3 Text Shadow Effect</title>

<style type="text/css">

h1 {

text-shadow: 5px 5px 10px #666;

}

h2 {

text-shadow: 5px 5px 10px red, 10px 10px 20px yellow;

}

</style>

</head>

<body>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

</body></html>

1. **2D Transformation**
2. **The Translate( ) Function**

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of CSS3 translate() Method</title>

<style type="text/css">

img {

-webkit-transform: translate(200px, 50px); /\* Chrome, Safari, Opera \*/

-moz-transform: translate(200px, 50px); /\* Firefox \*/

-ms-transform: translate(200px, 50px); /\* IE 9 \*/

-o-transform: translate(200px, 50px);/\* Opera \*/

transform: translate(200px, 50px); /\* Standard syntax \*/

}

.box{

margin: 50px;

width:153px;

height:103px;

background: url("images/tortoise-transparent.png") no-repeat;

}

</style>

</head>

<body>

<div class="box">

<img src="images/tortoise.png" alt="Tortoise">

</div>

</body></html>

1. **The Rotate ( ) Function**

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of CSS3 rotate() Method</title>

<style type="text/css">

img {

-webkit-transform: rotate(30deg); /\* Chrome, Safari, Opera \*/

-moz-transform: rotate(30deg); /\* Firefox \*/

-ms-transform: rotate(30deg); /\* IE 9 \*/

transform: rotate(30deg); /\* Modern Browsers \*/

}

.box{

margin: 50px;

width:120px;

height:110px;

background: url("images/star-fish-transparent.png") no-repeat;

}

</style>

</head>

<body>

<div class="box">

<img src="images/star-fish.png" alt="Star Fish">

</div>

</body></html>

1. **The Scale ( ) Function**

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of CSS3 scale() Method</title>

<style type="text/css">

img:hover {

transform: scale(1.5) rotate(200deg); /\* Modern Browsers \*/

opacity: 0.5;

}

.box{

margin: 50px;

width:103px;

height:130px;

background: url("images/octopus.png") no-repeat;

}

</style>

</head>

<body>

<div class="box">

<img src="images/octopus.png" alt="Octopus">

</div>

</body></html>

1. **3D Transformation**
2. **The Translate3d ( ) Function**

<html lang="en"><head>

<title>Example of CSS3 translate3d() Method</title>

<style type="text/css">

.container{

width: 125px;

height: 125px;

perspective: 500px;

border: 4px solid #e5a043;

background: #fff2dd;

margin: 30px;

}

.transformed {

-webkit-transform: translate3d(25px, 25px, 50px); /\* Chrome, Safari, Opera \*/

transform: translate3d(25px, 25px, 50px); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h2>Before Translation:</h2>

<div class="container">

<img src="images/diamond.jpg" alt="Diamond Card">

</div>

<h2>After Translation:</h2>

<div class="container">

<img src="images/diamond.jpg" class="transformed" alt="Diamond Card">

</div>

</body></html>

1. **The Rotate3d ( ) Function**

<html lang="en"><head>

<title>Example of CSS3 rotate3d() Method</title>

<style type="text/css">

.container{

margin: 50px;

width: 125px;

height: 125px;

perspective: 300px;

border: 4px solid #a2b058;

background: #f0f5d8;

}

.transformed {

transform: rotate3d(0, 1, 0, 60deg); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h2>Before Rotation:</h2>

<div class="container">

<img src="images/club.jpg" alt="Club Card">

</div>

<h2>After Rotation:</h2>

<div class="container">

<img src="images/club.jpg" class="transformed" alt="Club Card">

</div>

</body></html>

1. **The Scale 3d ( ) Function**

<html lang="en"><head>

<title>Example of CSS3 scale3d() Method</title>

<style type="text/css">

.container{

margin: 50px;

width: 125px;

height: 125px;

perspective: 300px;

border: 4px solid #6486ab;

background: #e9eef3;

}

.transformed {

-webkit-transform: scale3d(1, 1, 2) rotate3d(1, 0, 0, 60deg); /\* Chrome, Safari, Opera \*/

transform: scale3d(1, 1, 2) rotate3d(1, 0, 0, 60deg); /\* Standard syntax \*/

}

</style>

</head>

<body>

<h2>Before Scaling:</h2>

<div class="container">

<img src="images/spade.jpg" alt="Club Card">

</div>

<h2>After Scaling:</h2>

<div class="container">

<img src="images/spade.jpg" class="transformed" alt="Club Card">

</div>

</body></html>

1. **Transition**

Normally when the value of a CSS property changes, the rendered result is instantly updated. A common example is changing the background color of a button on mouse hover. In a normal scenario the background color of the button is changes immediately from the old property value to the new property value when you place the cursor over the button

**Example:**

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of CSS3 Multiple Transitions</title>

<style type="text/css">

button {

color: #fff;

padding: 10px 20px;

font: bold 18px sans-serif;

background: #fd7c2a;

border: 3px solid #dc5801;

/\* For Safari 3.0+ \*/

-webkit-transition-property: background, border;

-webkit-transition-duration: 1s, 2s;

/\* Standard syntax \*/

transition-property: background, border;

transition-duration:1s;

transition-delay:2s;

}

button:hover {

background: #3cc16e;

border-color: #288049;

}

</style>

</head>

<body>

<button type="button">Hover on me</button>

</body></html>

1. **Animation**

**Creating CSS3 Animations**

The CSS3 animations take it a step further with keyframe-based animations that allow you to specify the changes in CSS properties over time as a set of keyframes, like flash animations. Creating CSS animations is a two step process, as shown in the example below:

* The first step of building a CSS animation is to defining individual keyframes and naming an animation with a keyframes declaration.
* The second step is referencing the keyframes by name using the animation-name property as well as adding animation-duration and other optional animation properties to control the animation's behaviour.

However, it is not necessary to define the keyframes rules before referencing or applying it. The following example will show you how to animate a box horizontally from one position to another using the CSS3 animation feature.

**Example:**

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of CSS3 Translate Animation</title>

<style type="text/css">

.box {

margin: 50px;

width:153px;

height:103px;

background: url("images/tortoise-transparent.png") no-repeat;

position: relative;

/\* Chrome, Safari, Opera \*/

-webkit-animation-name: XYZ;

-webkit-animation-duration: 2s;

/\* Standard syntax \*/

animation-name: XYZ;

animation-duration: 2s;

}

/\* Chrome, Safari, Opera \*/

@-webkit-keyframes XYZ {

from {left: 0;}

to {left: 100%;}

}

/\* Standard syntax \*/

@keyframes XYZ {

from {left: 0;}

to {left: 90%;}

}

</style>

</head>

<body>

<div class="box"></div>

</body></html>

[**Defining Keyframes**](file:///E:\Pankaj\OFFICE\Training\Presentation\4.CSS3\Practice\CSS%203\8.2%20defining%20keyframe.html)

Keyframes are used to specify the values for the animating properties at various stages of the animation. Keyframes are specified using a specialized CSS at-rule — @keyframes. The keyframe selector for a keyframe style rule starts with a percentage (%) or the keywords from (same as 0%) or to (same as 100%). The selector is used to specify where a keyframe is constructed along the duration of the animation.

Percentages represent a percentage of the animation duration, 0% represents the starting point of the animation, 100% represents the end point, 50% represents the midpoint and so on. That means, a 50% keyframe in a 2s animation would be 1s into an animation.

**Animation Shorthand Property**

There are many properties to consider when creating the animations. However, it is also possible to specify all the animations properties in one single property to shorten the code.

The animation property is a shorthand property for setting all the individual animation properties at once in the listed order. For example:

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of CSS3 Shake Animation</title>

<style type="text/css">

.box {

margin: 50px;

width:120px;

height:110px;

background: url("images/star-fish-transparent.png") no-repeat;

position: relative;

left: 0;

/\* Chrome, Safari, Opera \*/

-webkit-animation-name: shakeit;

-webkit-animation-duration: 2s;

/\* Standard syntax \*/

animation-name: shakeit;

animation-duration: 2s;

}

/\* Chrome, Safari, Opera \*/

@-webkit-keyframes shakeit {

12.5% {left: -50px;}

25% {left: 50px;}

37.5% {left: -25px;}

50% {left: 25px;}

62.5% {left: -10px;}

75% {left: 10px;}

}

/\* Standard syntax \*/

@keyframes shakeit {

12.5% {left: -50px;}

25% {left: 50px;}

37.5% {left: -25px;}

50% {left: 25px;}

62.5% {left: -10px;}

75% {left: 10px;}

}

</style>

</head>

<body>

<p><strong>Note:</strong> Click the "Show Output" button to repeat the animation.</p>

<div class="box"></div>

</body></html>



1. Introduction
2. Advanced Input types
3. SVG Tag
4. Canvas Tag
5. Audio Tag
6. Video Tag
7. Geolocation
8. **Introduction**

New HTML 5 Elements

* New elements HEADER Tag, FOOTER Tag, ARTICLE Tag, and SECTION tag.
* New form control attributes like number, date, time, calendar, and range.
* New graphic elements: SVG Tag and CANVAS Tag
* New multimedia elements: AUDIO Tag and VIDEO Tag

New Semantic/Structural Elements

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <article> | Defines an article in the document |
| <aside> | Defines content aside from the page content |
| <bdi> | Defines a part of text that might be formatted in a different direction from other text |
| <details> | Defines additional details that the user can view or hide |
| <dialog> | Defines a dialog box or window |
| <figcaption> | Defines a caption for a <figure> element |
| <figure> | Defines self-contained content, like illustrations, diagrams, photos, code listings, etc. |
| <footer> | Defines a footer for the document or a section |
| <header> | Defines a header for the document or a section |
| <main> | Defines the main content of a document |
| <mark> | Defines marked or highlighted text |
| <menuitem> | Defines a command/menu item that the user can invoke from a popup menu |
| <meter> | Defines a scalar measurement within a known range (a gauge) |
| <nav> | Defines navigation links in the document |
| <progress> | Defines the progress of a task |
| <rp> | Defines what to show in browsers that do not support ruby annotations |
| <rt> | Defines an explanation/pronunciation of characters (for East Asian typography) |
| <ruby> | Defines a ruby annotation (for East Asian typography) |
| <section> | Defines a section in the document |
| <summary> | Defines a visible heading for a <details> element |
| <time> | Defines a date/time |
| <wbr> | Defines a possible line-break |

New Input Types

|  |  |
| --- | --- |
| **New Input Types** | **New Input Attributes** |
| color  date  datetime  datetime-local  email  month  number  range  search  tel  time  url  week | Autocomplete  autofocus  form  formaction  formenctype  formmethod  formnovalidate  formtarget  height and width  list  min and max  multiple  pattern (regexp)  placeholder  required  step |

HTML5 Graphics

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <canvas> | Defines graphic drawing using JavaScript |
| <svg> | Defines graphic drawing using SVG |

New Media Elements

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <audio> | Defines sound or music content |
| <embed> | Defines containers for external applications (like plug-ins) |
| <source> | Defines sources for <video> and <audio> |
| <track> | Defines tracks for <video> and <audio> |
| <video> | Defines video or movie content |

HTML5 Browser Support

HTML5 is supported in all modern browsers.

Define HTML5 Elements as Block Elements

* HTML5 defines eight new semantic HTML elements. All these are block-level elements.
* header, section, footer, aside, nav, main, article, figure
* You can also add any new element to HTML

Example:-

<html>

<head>

<script type="text/javascript">

document.createElement("myHero");

document.createElement("pankaj");

document.createElement("bipin");

</script>

<style>

myHero {

display: block;

background-color: #ddd;

padding: 50px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

<myhero>My First Hero</myhero>

<pankaj>pankaj</pankaj>

<bipin>Bipin</bipin>

</body></html>

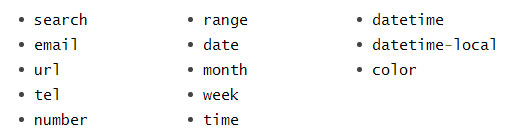
What are Semantic Elements?

* A semantic element clearly describes its meaning to both the browser and the developer.
* Examples of non-semantic elements: DIV tag and SPAN tag- Tells nothing about its content.
* Examples of semantic elements: FORM tag, TABLE tag, and IMG tag - Clearly defines its content.



1. **Advanced Input Type**

New Input types are as below



Example :

<html xmlns="http://www.w3.org/1999/xhtml"><head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Untitled Document</title>

<style type="text/css">

.field-wrpr {

float: left;

width: 100%;

}

</style>

</head>

<body>

<form>

<div class="field-wrpr">

<h2> type Search</h2>

<label> Search Website:

<input type="search" name="mysearch">

</label>

</div>

<div class="field-wrpr">

<h2> type Range</h2>

<label> Select Number:

<input type="range" value="1" min="1" max="10" step="0.5" name="mynumber">

</label>

</div>

<div class="field-wrpr">

<h2> type Date Time</h2>

<label> Date &amp; Time:

<input type="date" name="mydatetime">

</label>

</div>

<div class="field-wrpr">

<h2> type Email</h2>

<label>

Email Address: <input type="email" required="">

<button type="submit">Save</button>

</label>

</div>

<div class="field-wrpr">

<h2> type Date</h2>

<label>

Select Date: <input type="date" name="mydate">

</label>

</div>

<div class="field-wrpr">

<h2> type Local Date &amp; Time</h2>

<label>

Local Date &amp; Time: <input type="datetime-local" name="mylocaldatetime">

</label>

</div>

<div class="field-wrpr">

<h2> type URL</h2>

<label>

Website URL: <input type="url" name="mywebsite" required="">

</label>

</div>

<div class="field-wrpr">

<h2> type URL</h2>

<label>

Select Month: <input type="month" name="mymonth">

</label>

</div>

<div class="field-wrpr">

<h2> type Color</h2>

<label>

Select Color: <input type="color" name="mycolor">

</label>

</div>

<div class="field-wrpr">

<h2> type tel</h2>

<label>

Telephone Number: <input type="tel" name="mytelephone" required="">

</label>

</div>

<div class="field-wrpr">

<h2> type Week</h2>

<label>

Select Week: <input type="week" name="myweek">

</label>

</div>

<div class="field-wrpr">

<h2> type Number</h2>

<label> Select Number:

<input type="number" value="6" min="1.5" max="10" step="1.5" name="mynumber">

</label>

<p><strong>Note</strong>: If you try to enter the number out of the range (1-10) or text character it will show error.</p>

</div>

</form>

</body></html>

1. **SVG**

What is SVG?

* SVG stands for Scalable Vector Graphics
* SVG is used to define vector-based graphics for the Web
* SVG defines the graphics in XML format
* SVG graphics do NOT lose any quality if they are zoomed or resized
* Every element and every attribute in SVG files can be animated
* SVG is a W3C recommendation
* SVG integrates with other W3C standards such as the DOM and XSL

SVG Advantages

* SVG images can be created and edited with any text editor
* SVG images can be searched, indexed, scripted, and compressed
* SVG images are scalable
* SVG images can be printed with high quality at any resolution
* SVG images are zoom able (and the image can be zoomed without degradation)
* SVG is an open standard
* SVG files are pure XML

SVG Example

* Start With svg
* Line Using SVG
* Rectangle Using SVG
* Circle Using SVG
* Text Using SVG
* Transform Using SVG

1. Start With svg

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Embedding SVG Into HTML Pages</title>

<style type="text/css">

svg{border:1px solid #000}

</style>

</head>

<body>

<svg width="600" height="200">

<text x="10" y="20" style="font-size:14px; color:#ccc">

Your browser support SVG.

</text>

Sorry, your browser does not support SVG.

</svg>

</body>

</html>

1. Line Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Create a Line with HTML5 SVG</title>

<style type="text/css">

svg {border: 1px solid black; }

</style>

</head>

<body>

<svg width="300" height="200">

<line x1="10" y1="10" x2="150" y2="100" style="stroke:#999; stroke-width:1;"></line>

</svg>

</body></html>

1. Rectangle Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Create a Rectangle with HTML5 SVG</title>

<style type="text/css">

svg {border: 1px solid black;}

</style>

</head>

<body>

<svg width="300" height="200">

<rect x="50" y="50" width="200" height="100" style="fill:red; stroke:yellow; stroke-width:5;"></rect>

</svg>

</body></html>

1. Circle Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Create a Circle with HTML5 SVG</title>

<style type="text/css">

svg {border: 1px solid black;}

</style>

</head>

<body>

<svg width="300" height="200">

<circle cx="150" cy="100" r="100" style="fill:green; stroke:red; stroke-width:1;"></circle>

</svg>

</body></html>

1. Text Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Render Text with HTML5 SVG</title>

<style type="text/css">

svg {border: 1px solid black; }

</style>

</head>

<body>

<svg width="300" height="200">

<text x="20" y="30" style="fill:purple; font-size:22px; transform:rotate(10deg);">

Welcome to Our Website!

</text>

<text x="20" y="30" dx="0" dy="20" style="fill:navy; font-size:14px; transform:rotate(45deg);">

Here you will find lots of useful information.

</text>

</svg>

</body></html>

1. Transform Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Rotate and Render Text with HTML5 SVG</title>

<style type="text/css">

svg {border: 1px solid black;}

</style>

</head>

<body>

<svg width="300" height="200">

<text x="30" y="15" style="fill:purple; font-size:22px; transform:rotate(45deg);">

<tspan style="fill:purple; font-size:22px;">

Welcome to Our Website!

</tspan>

<tspan dx="-230" dy="20" style="fill:navy; font-size:14px;">

Here you will find lots of useful information.

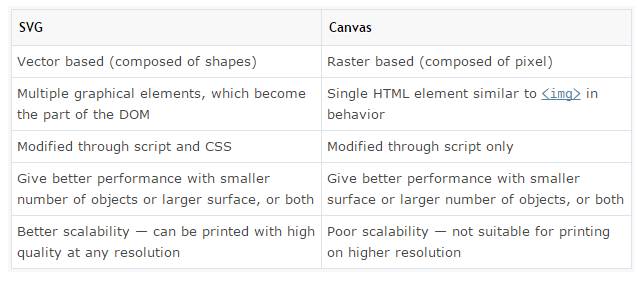
</tspan>

</text>

</svg>

</body></html>

1. **Canvas Tag**

****

What is HTML Canvas?

* The HTML CANVAS element is used to draw graphics, on the fly, via scripting (usually JavaScript).
* The CANVAS element is only a container for graphics. You must use a script to actually draw the graphics.
* Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Example:

* Inserting CANVAS tag
* Line Using Canvas
* ARC Using Canvas
* Rectangle Using canvas
* Stroke Using canvas
* Cap Stroke Using canvas
* Inside a rectangle
* Inside a circle

1. Inserting CANVAS tag

<html lang="en"><head>

<meta charset="UTF-8">

<title>HTML5 Canvas</title>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

// draw stuff here

};

</script>

<style>

canvas{border:1px solid #000}

</style>

</head>

<body>

<canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Line Using Canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Drawing a Line on Canvas</title>

<style type="text/css">

canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.moveTo(20, 20);

context.lineTo(100, 150);

context.lineTo(300, 300);

context.lineTo(75, 100);

context.stroke();

};

</script>

</head>

<body>

<canvas id="myCanvas" width="700" height="400"></canvas>

</body></html>

1. ARC Using Canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Drawing an Arc on Canvas</title>

<style type="text/css">

canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

//context.arc(centerX, centerY, radius, startingAngle, endingAngle, counterclockwise);

context.arc(150, 150, 50, 0 \* Math.PI, 0.5 \* Math.PI, true);

context.stroke();

};

</script>

</head>

<body>

<canvas id="myCanvas" width="500" height="500"></canvas>

</body></html>

1. Rectangle Using canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Drawing a Rectangle on Canvas</title>

<style type="text/css">

canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.lineWidth = 2;

context.strokeStyle = "red";

//context.rect(x, y, width, height);

context.rect(50, 50, 200, 200);

context.stroke();

};

</script>

</head>

<body>

<canvas id="myCanvas" width="300" height="300"></canvas>

</body></html>

1. Stroke Using canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Setting Stroke Color and Width</title>

<style type="text/css">

canvas{border: 1px solid #000; }

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.lineWidth = 5;

context.strokeStyle = "blue";

context.moveTo(50, 150);

context.lineTo(250, 50);

context.stroke();

context.lineCap = "round";

};

</script>

</head>

<body>

<canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Cap Stroke Using canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Setting Stroke Cap Style</title>

<style type="text/css">

canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.lineWidth = 10;

context.strokeStyle = "orange";

//butt, round, and square.

context.lineCap = "butt";

context.arc(150, 150, 80, 1.2 \* Math.PI, 1.8 \* Math.PI, false);

context.stroke();

};

</script>

</head>

<body>

<canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Inside a rectangle

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Filling Color inside a Rectangle</title>

<style type="text/css">

canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.rect(50, 50, 200, 100);

context.fillStyle = "#999";

//It is recommended to use the fill() method before the stroke() method in order to render the stroke correctly.

context.fill();

context.lineWidth = 5;

context.strokeStyle = "blue";

context.stroke();

};

</script>

</head>

<body>

<canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Inside a circle

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Filling Color inside a Circle</title>

<style type="text/css">

canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

window.onload = function(){

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.arc(150, 100, 70, 0, 2 \* Math.PI, false);

context.fillStyle = "#FB8B89";

//It is recommended to use the fill() method before the stroke() method in order to render the stroke correctly.

context.fill();

context.lineWidth = 5;

context.strokeStyle = "black";

context.stroke();

};

</script>

</head>

<body>

<canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

**5) Audio Tag**

* The audio tag defines sound, such as music or other audio streams.
* Currently, there are 3 supported file formats for the audio tag element: MP3, Wav, and Ogg

Example :

<html lang="en"><head>

<meta charset="UTF-8">

<title>The HTML5 audio Element</title>

</head>

<body>

<audio control autoplay="true" src="6.MUSIC/Kalimba.mp3” >

Your browser does not support the HTML5 audio element.

</audio>

</body></html>

**5) Video Tag**

* Before HTML5, there was no standard for showing videos on a web page.
* Before HTML5, videos could only be played with a plug-in (like flash).
* The HTML5 video tag element specifies a standard way to embed a video in a web page.

Example :

<html lang="en"><head>

<meta charset="UTF-8">

<title>The HTML5 video element</title>

<style>

video{

position:relative;

}

.video-text p{

position:relative;top:100px; font-size:30px;font-weight:bold;color:#fff

}

</style></head>

<body>

<div class="video-text">

<video autoplay="" controls="" src="7.VIDEO/video.mkv">

Your browser does not support the HTML5 Video element.

</video>

<p>Hello</p>

</div>

</body></html>

1. **Geolocation**

The HTML5 geolocation feature lets you find out the geographic coordinates (latitude and longitude numbers) of the current location of your website's visitor. This feature is helpful for providing better browsing experience to the site visitor. For example, you can return the search results that are physically close to the user's location.

Example:

* GETTING CO-ORDINATES
* HANDELING ERROR AND REJECTION
* SHOW GEOLOCATION AS IMAGE
* GELOCATION BY GOOGLE MAP
* CURRENT LOCATION OF VISITOR

1. GETTING CO-ORDINATES

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of HTML5 Geolocation</title>

<script type="text/javascript">

function showPosition(){

if(navigator.geolocation){

navigator.geolocation.getCurrentPosition(function(position){

var positionInfo = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";

document.getElementById("result").innerHTML = positionInfo;

});

} else{

alert("Sorry, your browser does not support HTML5 geolocation.");

}

}

</script>

</head>

<body>

<div id="result">

<!--Position information will be inserted here-->

</div>

<button type="button" onclick="showPosition();">Show Position</button>

</body></html>

1. HANDELING ERROR AND REJECTION

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Handling Geolocation Errors and Rejections</title>

<script type="text/javascript">

// Set up global variable

var result;

function showPosition(){

// Store the element where the page displays the result

result = document.getElementById("result");

// If geolocation is available, try to get the visitor's position

if(navigator.geolocation){

navigator.geolocation.getCurrentPosition(successCallback, errorCallback);

result.innerHTML = "Getting the position information...";

} else{

alert("Sorry, your browser does not support HTML5 geolocation.");

}

};

// Define callback function for successful attempt

function successCallback(position){

result.innerHTML = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";

}

// Define callback function for failed attempt

function errorCallback(error){

if(error.code == 1){

result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";

} else if(error.code == 2){

result.innerHTML = "The network is down or the positioning service can't be reached.";

} else if(error.code == 3){

result.innerHTML = "The attempt timed out before it could get the location data.";

} else{

result.innerHTML = "Geolocation failed due to unknown error.";

}

}

</script>

</head>

<body>

<div id="result">

<!--Position information will be inserted here-->

</div>

<button type="button" onclick="showPosition();">Show Position</button>

</body></html>

1. SHOW GEOLOCATION AS IMAGE

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Showing Geolocation on Google Map</title>

<script type="text/javascript">

function showPosition(){

navigator.geolocation.getCurrentPosition(showMap);

}

function showMap(position){

// Get location data

var latlong = position.coords.latitude + "," + position.coords.longitude;

// Set Google map source url

var mapLink = "http://maps.googleapis.com/maps/api/staticmap?center="+latlong+"&zoom=16&size=400x300&output=embed";

// Create and insert Google map

document.getElementById("embedMap").innerHTML = "<img alt='Map Holder' src='"+ mapLink +"'>";

}

</script>

</head>

<body>

<button type="button" onclick="showPosition();">Show My Position on Google Map</button>

<div id="embedMap">

<!--Google map will be embedded here-->

</div>

</body></html>

1. GELOCATION BY GOOGLE MAP

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Showing Location on Google Map</title>

<script src="http://maps.google.com/maps/api/js?sensor=false"></script>

<script type="text/javascript">

function showPosition(){

if(navigator.geolocation){

navigator.geolocation.getCurrentPosition(showMap, showError);

} else{

alert("Sorry, your browser does not support HTML5 geolocation.");

}

}

// Define callback function for successful attempt

function showMap(position){

// Get location data

lat = position.coords.latitude;

long = position.coords.longitude;

var latlong = new google.maps.LatLng(lat, long);

var myOptions = {

center: latlong,

zoom: 16,

mapTypeControl: true,

navigationControlOptions: {style:google.maps.NavigationControlStyle.SMALL}

}

var map = new google.maps.Map(document.getElementById("embedMap"), myOptions);

var marker = new google.maps.Marker({position:latlong, map:map, title:"You are here!"});

}

// Define callback function for failed attempt

function showError(error){

if(error.code == 1){

result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";

} else if(error.code == 2){

result.innerHTML = "The network is down or the positioning service can't be reached.";

} else if(error.code == 3){

result.innerHTML = "The attempt timed out before it could get the location data.";

} else{

result.innerHTML = "Geolocation failed due to unknown error.";

}

}

</script>

</head>

<body>

<button type="button" onclick="showPosition();">Show My Position on Google Map</button>

<div id="embedMap" style="width: 400px; height: 300px;">

<!--Google map will be embedded here-->

</div>

</body></html>

1. CURRENT LOCATION OF VISITOR

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Getting Current Position</title>

<script type="text/javascript">

// Set global variable

var watchID;

function showPosition(){

if(navigator.geolocation){

watchID = navigator.geolocation.watchPosition(successCallback);

} else{

alert("Sorry, your browser does not support HTML5 geolocation.");

}

}

function successCallback(position){

toggleWatchBtn.innerHTML = "Stop Watching";

// Check position has been changed or not before doing anything

if(prevLat != position.coords.latitude || prevLong != position.coords.longitude){

// Set previous location

var prevLat = position.coords.latitude;

var prevLong = position.coords.longitude;

// Get current position

var positionInfo = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";

document.getElementById("result").innerHTML = positionInfo;

}

}

function startWatch(){

var result = document.getElementById("result");

var toggleWatchBtn = document.getElementById("toggleWatchBtn");

toggleWatchBtn.onclick = function(){

if(watchID){

toggleWatchBtn.innerHTML = "Start Watching";

navigator.geolocation.clearWatch(watchID);

watchID = false;

}

else{

toggleWatchBtn.innerHTML = "Aquiring Geo Location...";

showPosition();

}

}

}

// Initialise the whole system (above)

window.onload = startWatch;

</script>

</head>

<body>

<button type="button" id="toggleWatchBtn">Start Watching</button>

<div id="result">

<!--Position information will be inserted here-->

</div>

</body></html>



**Syllabus for JavaScript**

1. Introduction
2. Syntax
3. Statement and Comment
4. Variable
5. Data types
6. Operator
7. String
8. Event
9. Alert
10. Loop statement
11. Switch statement
12. While and do while Statement
13. For Loop statement
14. Function
15. Page Redirect
16. Page Printing

**1. Introduction**

**What JAVA is?**

Java is a programming language developed in 1995 for Sun Microsystems. Its main purpose is to be able to run on all different types of operating systems. In other words, a program written in Java can run on a Microsoft computer and an Apple computer. All you need is to download the Java run-time to your computer and you can run any Java applications.

Java can be used on the Web, but I haven’t seen an example of a site using Java in a long time. If someone is telling you they’ll build your website using Java, they probably mean JavaScript.

**What JAVASCRIPT is?**

JavaScript is built to run on the Web. How it executes and how well it performs is dependent on the type of browser you are using (Check out this website to help you determine: What is a browser?).

JavaScript was decried, denigrated and dismissed by many programmers for years, including myself. The relatively recent rise of “fancy” websites that update quickly without reloading a page (Ajax or asynchronous JavaScript execution) has brought JavaScript into prominence, and has even made an old hater like me at least acknowledge its importance.

If a salesperson is talking to you about using JavaScript to build something on your website, they are probably talking about implementing some sort of “fancy” interface feature to make the site more appealing to your visitors.

**What is JQuery?**

jQuery is what made it palatable and fun to start using JavaScript again. It is the most popular JavaScript library in use today, and if you are building a site you should be using it. If you are having a site built, when the sales person talks about JavaScript, it is almost certain that the developer will be utilizing jQuery to make it happen.

**2. Syntax**

JavaScript can be implemented using JavaScript statements that are placed within the script... script HTML tags in a web page.

You can place the script tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the head tags.

<script type=”text/javascript”>

Document.write (“Hello World”);

</script>

**3. Statement & Comment**

// Single line comment

/\* Multiple Line Comment \*/

**4. Variable**

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

* **Numbers** e.g. 123, 120.50 etc.
* **Strings** e.g. "This text string" etc.
* **Boolean** e.g. true or false.

Java does not make a distinction between integer values and floating-point values. All numbers in JavaScript are represented as floating-point values. JavaScript represents numbers using the 64-bit floating-point format defined by the IEEE 754 standard.

<script type=”text/javascript”>

var x=50;

var y=25;

var z=30;

document.write(x);

document.write('<br>');

document.write(y);

document.write('<br>');

document.write(z);

</script>

**5. Data Types**

<script type="text/javascript">

//integer

var myvariable = 20;

document.write(myvariable);

//float

var myvariable\_float = 30.95;

document.write(myvariable\_float);

//string

var myvariable\_string = 'what is your name? Answer is “my name is PANKAJ"';

document.write(myvariable\_string);

</script>

**6. Operator**

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and ‘+’ is called the operator. JavaScript supports the following types of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

1. **Arithmetic Operators**

JavaScript supports the following arithmetic operators − Assume variable A holds 10 and variable B holds 20, then –

* 1. + Addition = Adds two operands Ex: A + B will give 30
  2. – Subtraction = Subtracts the second operand from the first Ex: A - B will give -10
  3. \* Multiplication = Multiply both operands Ex: A \* B will give 200
  4. / Division = Divide the numerator by the denominator Ex: B / A will give 2
  5. % Modulus = Outputs the remainder of an integer division Ex: B % A will give 0
  6. ++ Increment = Increases an integer value by one Ex: A++ will give 11
  7. -- Decrement = Decreases an integer value by one Ex: A-- will give 9

**Example : -**

<html>

<body>

<script type=”text/Javascript”>

var a = 33, b = 10, c = "Test", linebreak = "<br/>";

document.write("a + b = "); result = a + b; document.write(result); document.write(linebreak);

document.write("a - b = "); result = a - b; document.write(result); document.write(linebreak);

document.write("a / b = "); result = a / b; document.write(result); document.write(linebreak);

document.write("a % b = ");result = a % b; document.write(result); document.write(linebreak);

document.write("a + b + c = "); result = a + b + c; document.write(result); document.write(linebreak);

a = a++;document.write("a++ = "); result = a++; document.write(result); document.write(linebreak);

b = b--; document.write("b-- = "); result = b--; document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

a + b = 43

a - b = 23

a / b = 3.3

a % b = 3

a + b + c = 43Test

a++ = 33

b-- = 10

1. Comparison Operators

JavaScript supports the following comparison operators − Assume variable A holds 10 and variable B holds 20, then −

1. = = Equal Checks if the value of two operands are equal or not, if yes, then the condition becomes true. Ex: A == B is not true.
2. != Not Equal Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: A! = B is true.
3. > Greater than Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: A > B is not true.
4. < Less than Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: A < B is true.
5. >= Greater than or Equal to Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: A >= B is not true.
6. <= Less than or Equal to Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: A <= B is true.

**Example : -**

<html>

<body>

<script type=”text/Javascript”>

var a = 10, b = 20, linebreak = "<br />";

document.write("(a == b) => "); result = (a == b); document.write(result); document.write(linebreak);

document.write("(a < b) => "); result = (a < b); document.write(result); document.write(linebreak);

document.write("(a > b) => "); result = (a > b); document.write(result); document.write(linebreak);

document.write("(a != b) => "); result = (a != b); document.write(result); document.write(linebreak);

document.write("(a >= b) => ");result = (a >= b); document.write(result)document.write(linebreak);

document.write("(a <= b) => "); result = (a <= b);document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

(a == b) => false

(a < b) => true

(a > b) => false

(a != b) => true

(a >= b) => false

a <= b) => true

1. **Logical Operator**
   1. && Logical AND If both the operands are non-zero, then the condition becomes true. Ex: A && B is true.
   2. || Logical OR If any of the two operands are non-zero, then the condition becomes true. Ex: A | | B is true.
   3. ! Logical NOT Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. Ex: ! A && B is false

<html>

<body>

<script type=”text/Javascript”>

var a = true; var b = false; var linebreak = "<br />";

document.write("(a && b) => "); result = (a && b); document.write(result); document.write(linebreak);

document.write("(a || b) => "); result = (a || b); document.write(result); document.write(linebreak);

document.write("!(a && b) => "); result = (!(a && b)); document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

(a && b) => false

(a || b) => true

!(a && b) => true

1. **Assignment Operator**
2. = Simple Assignment Assigns values from the right side operand to the left side operand Ex: C = A + B will assign the value of A + B into C
3. += Add and Assignment It adds the right operand to the left operand and assigns the result to the left operand. Ex: C += A is equivalent to C = C + A
4. −= Subtract and Assignment It subtracts the right operand from the left operand and assigns the result to the left operand. Ex: C -= A is equivalent to C = C – A
5. \*= Multiply and Assignment It multiplies the right operand with the left operand and assigns the result to the left operand. Ex: C \*= A is equivalent to C = C \* A
6. /= Divide and Assignment It divides the left operand with the right operand and assigns the result to the left operand. Ex: C /= A is equivalent to C = C / A
7. %= Modules and Assignment It takes modulus using two operands and assigns the result to the left operand. Ex: C %= A is equivalent to C = C % A

<html>

<body>

<script type=”text/Javascript”>

var a = 33; var b = 10; var linebreak = "<br />";

document.write("Value of a => (a = b) => "); result = (a = b); document.write(result); document.write(linebreak);

document.write("Value of a => (a += b) => "); result = (a += b); document.write(result); document.write(linebreak);

document.write("Value of a => (a -= b) => "); result = (a -= b); document.write(result). document.write(linebreak);

document.write("Value of a => (a \*= b) => "); result = (a \*= b); document.write(result); document.write(linebreak);

document.write("Value of a => (a /= b) => "); result = (a /= b); document.write(result); document.write(linebreak);

document.write("Value of a => (a %= b) => "); result = (a %= b); document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

Value of a => (a = b) => 10

Value of a => (a += b) => 20

Value of a => (a -= b) => 10

Value of a => (a \*= b) => 100

Value of a => (a /= b) => 10

Value of a => (a %= b) => 0

1. **Conditional Operators**

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation. –

* 1. ? : Conditional If Condition is true? Then value X : Otherwise value Y

**Example:-**

<html>

<body>

<script type=”text/Javascript”>

var a = 10; var b = 20; var linebreak = "<br />";

document.write ("((a > b) ? 100 : 200) => "); result = (a > b) ? 100 : 200; document.write(result); document.write(linebreak);

document.write ("((a < b) ? 100 : 200) => "); result = (a < b) ? 100 : 200; document.write(result); document.write(linebreak);

</script>

</body>

</html>

**OUTPUT**

((a > b) ? 100 : 200) => 200

((a < b) ? 100 : 200) => 100

**6. String**

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods. As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

**Syntax**

Use the following syntax to create a String object –

var val = new String(string);

**Example :-**

<script type="text/javascript">

var name = "PANKAJ";

// escape charecter " \ "

//var massage = "Hello world,\" I am very Happy\"";

//document.write(massage);

// concantation

var sentance1 = " Hello World";

var sentance2 = " I am Very Happy ";

document.write(sentance1 + sentance2 );

// concantation with number

//var x =20;

//document.write('my age is ' + x);

// HTML in sring

document.write("<h1> This is Heading in string</h1> ");

</script>

**8. Alert**

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and ‘+’ is called the operator. JavaScript supports the following types of operators.

* Alert On Page Load
* Alert On Button Click
* Alert On Confirmation Box
* Alert On Promot Box

1. **Alert On Load**

<html>

<body>

<script type="text/javascript">

alert("ENTER TEXT")

</script>

</body>

</html>

1. **Alert On Button Click**

<html>

<body>

<!--alert On page load -->

<button onmouseover="myFunction()">Click Me</button>

<script>

function myFunction() {

alert("Alert on function call");

}

</script>

</body>

</html>

1. **Alert On Confirmation Box**

<html>

<body>

<!-- Confirmation Box on click -->

<button onclick="Confirm()">Click for fonfirmation box</button>

<p id="demo"></p>

<p>without demo id</p>

<script>

function Confirm() {

var x;

if (confirm("Press a button!") == true) {

x = "You pressed OK!";

} else {

x = "You pressed Cancel!";

}

document.getElementById("demo").innerHTML = x;

}

</script>

</body>

</html>

1. **Alert On Promot Box**

<html>

<body>

<!--Promt Box -->

<button onclick="Prompt()">Prompt</button>

<p id="demo"></p>

<script>

function Prompt() {

var person = prompt("", "Please Enter Your Age");

if (person != null) {

document.getElementById("demo").innerHTML =

"Hello " + person + "! How are you today?";

}

}

</script>

</body>

</html>

**9. Event**

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable. Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

## Input Events

[onblur - When a user leaves an input field](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onblur)  
[onchange - When a user changes the content of an input field](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onchange)  
[onchange - When a user selects a dropdown value](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_dropdown)  
[onfocus - When an input field gets focus](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onfocus)  
[onselect - When input text is selected](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onselect)  
[onsubmit - When a user clicks the submit button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onsubmit)  
[onreset - When a user clicks the reset button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onreset)  
[onkeydown - When a user is pressing/holding down a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeydown)  
[onkeypress - When a user is pressing/holding down a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeypress)  
[onkeyup - When the user releases a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeyup)  
[onkeyup - When the user releases a key](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeyup2)  
[onkeydown vs onkeyup - Both](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onkeydownup)

**Mouse Events**

[onmouseover/onmouseout - When the mouse passes over an element](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmouse)  
[onmousedown/onmouseup - When pressing/releasing a mouse button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmousedown)  
[onmousedown - When mouse is clicked: Alert which element](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_srcelement)  
[onmousedown - When mouse is clicked: Alert which button](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmousedown2)  
[onmousemove/onmouseout - When moving the mouse pointer over/out of an image](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmousemove)  
[onmouseover/onmouseout - When moving the mouse over/out of an image](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmouseover)  
[onmouseover an image map](http://www.w3schools.com/js/tryit.asp?filename=tryjs_imagemap)

**Click Events**

[Acting to the onclick event](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events)  
[onclick - When button is clicked](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onclick)  
[ondblclick - When a text is double-clicked](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_ondblclick)

**Load Events**

[onload - When the page has been loaded](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_body_onload)  
[onload - When an image has been loaded](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_img_onload)  
[onerror - When an error occurs when loading an image](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onerror)  
[onunload - When the browser closes the document](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onunload)  
[onresize - When the browser window is resized](http://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onresize)

**10. Loop Statement**

1. If Statement
2. If .. Else Statement
3. If .. Else .. If Statement
4. **If Statement**

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

If (expression) {

Statement(s) to be executed if expression is true

}

Example :-

<html>

<body>

<p id="demo">Age must be above 18</p>

<p>Set the variable to different value and then try...</p>

<script type="text/javascript">

var age = 10;

if( age > 18 ){ document.getElementById("demo").innerHTML = 'Eligible For Licences';

}

</script>

</body>

</html>

1. **If .. Else Statement**

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

If (expression) {

Statement(s) to be executed if expression is true

}

Else{

State ment(s) to be executed if expression is False

}

Example :-

<html>

<body>

<p id="demo">Age must be above 18</p>

<p>Set the variable to different value and then try...</p>

<script type="text/javascript">

var age = 15;

if( age > 18 ){

document.write("<b>Qualifies for driving</b>");

}

else{

document.write("<b>Does not qualify for driving</b>");

}

</script>

</body>

</html>

1. **If .. Else If.. Statement**

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

Syntax

If (expression 1) {

Statement(s) to be executed if expression1 is true

}

Else If (expression 2) {

Statement(s) to be executed if expression2 is true

}

Else if If (expression 3) {

Statement(s) to be executed if expression 3 is true

}

Else{

State ment(s) to be executed if all expression is False

}

Example :-

<html>

<body>

<p id="demo">Age must be above 18</p>

<p>Set the variable to different value and then try...</p>

<script type="text/javascript">

var book = "history";

var librarian = 20 + book;

if( librarian === 20 + book ){

document.write("<b>Librainan Msg Book</b>");

}

else if( book == "maths" ){

document.write("<b>Maths Book</b>");

}

else if( book == "economics" ){

document.write("<b>Economics Book</b>");

}

else{

document.write("<b>Unknown Book</b>");

}

</script>

</body>

</html>

**11. Switch Statement**

You can use multiple if...else…if statements, as in the previous chapter, to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.

Starting with JavaScript 1.2, you can use a switch statement which handles exactly this situation, and it does so more efficiently than repeated if...else if statements.

Syntax

switch (expression ) {

case condition 1 : statement(s);

break;

case condition 2 : statement(s);

break;

case condition 3 : statement(s);

break;

default : Statement(s);

}

Example :-

<html>

<body>

<script type="text/javascript">

var grade='B';

document.write("Entering switch block<br />");

switch (grade)

{

case 'A': document.write("Good job<br />");

break;

case 'B': document.write("Pretty good<br />");

break;

case 'C': document.write("Passed<br />");

break;

case 'D': document.write("Not so good<br />");

break;

case 'F': document.write("Failed<br />");

break;

default: document.write("Unknown grade<br />")

}

document.write("Exiting switch block");

</script>

</body>

</html>

**12. While and Do while Statement**

While writing a program, you may encounter a situation where you need to perform an action over and over again. In such situations, you would need to write loop statements to reduce the number of lines.

JavaScript supports the following forms of While Loop –

1. While Loop
2. Do..While
3. **While Loop**

The most basic loop in JavaScript is the **while** loop which would be discussed in this chapter. The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Syntax

While (expression 1) {

Statement(s) to be executed if expression1 is true

}

Example :-

<html>

<body>

<script type="text/javascript">

var count = 0;

document.write("Starting Loop ");

while (count < 9){

document.write("Current Count : " + count + "<br />");

count++;

}

document.write("Loop stopped!");

</script>

</body>

</html>

Output

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Loop stopped!

1. **Do ... While Loop**

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

Syntax

do{

Statement(s) to be executed;

} while (expression);

Example :-

<html>

<body>

<script type="text/javascript">

var count = 0;

document.write("Starting Loop" + "<br />");

do{

document.write("Current Count : " + count + "<br />");

count++;

}

while (count < 5);

document.write ("Loop stopped!");

</script>

</body>

</html>

**Output**

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Loop Stopped!

**13. For Loop**

The '**for**' loop is the most compact form of looping. It includes the following three important parts −

* The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
* The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
* The **iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

### Syntax

for (initialization; test condition; iteration statement){

Statement(s) to be executed if test condition is true

}

### Example

Try the following example to learn how a for loop works in JavaScript.

<html>

<body>

<script type="text/javascript">

var count;

document.write("Starting Loop" + "<br />");

for(count = 0; count < 10; count++){

document.write("Current Count : " + count );

document.write("<br />");

}

document.write("Loop stopped!");

</script>

</body>

</html>

**Output**

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped!

**14. Function**

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

**Syntax**

The basic syntax is shown here.

<script type="text/javascript">

function functionname(parameter-list)

{

statements

}

</script>

**Example**

Try the following example. It defines a function called sayHello that takes no parameters −

<script type="text/javascript">

function sayHello()

{

alert("Hello there");

}

</script>

**Calling a Function**

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

<html>

<head>

<script type="text/javascript">

function sayHello()

{

document.write ("Hello there!");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="sayHello()" value="Say Hello">

</form>

<p>Use different text in write method and then try...</p>

</body>

</html>

**Output**

**Function**

**Parameters**

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

**Example**

Try the following example. We have modified our **sayHello** function here. Now it takes two parameters.

<html>

<head>

<script type="text/javascript">

function sayHello(name, age)

{

document.write (name + " is " + age + " years old.");

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">

</form>

<p>Use different parameters inside the function and then try...</p>

</body>

</html>

**The return Statement**

A JavaScript function can have an optional **return** statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

**Example**

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

<html>

<head>

<script type="text/javascript">

function concatenate(first, last)

{

var full;

full = first + last;

return full;

}

function secondFunction()

{

var result;

result = concatenate('Zara', 'Ali');

document.write (result );

}

</script>

</head>

<body>

<p>Click the following button to call the function</p>

<form>

<input type="button" onclick="secondFunction()" value="Call Function">

</form>

<p>Use different parameters inside the function and then try...</p>

</body>

</html>

## 14. Page Redirection

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Example 1 :-

<html>

<head>

<script type="text/javascript">

function Redirect() {

window.location="http://www. enosislearning.com";

}

</script>

</head>

<body>

<p>Click the following button, you will be redirected to home page.</p>

<form>

<input type="button" value="Redirect Me" onclick="Redirect();" />

</form>

</body>

</html>

Example 2 :-

<html>

<head>

<script type="text/javascript">

function Redirect() {

window.location="http://www.enosislearning.com";

}

document.write("You will be redirected to main page in 10 sec.");

setTimeout('Redirect()', 10000);

</script>

</head>

<body>

</body>

</html>